Introduction Of CNN Deep Learning

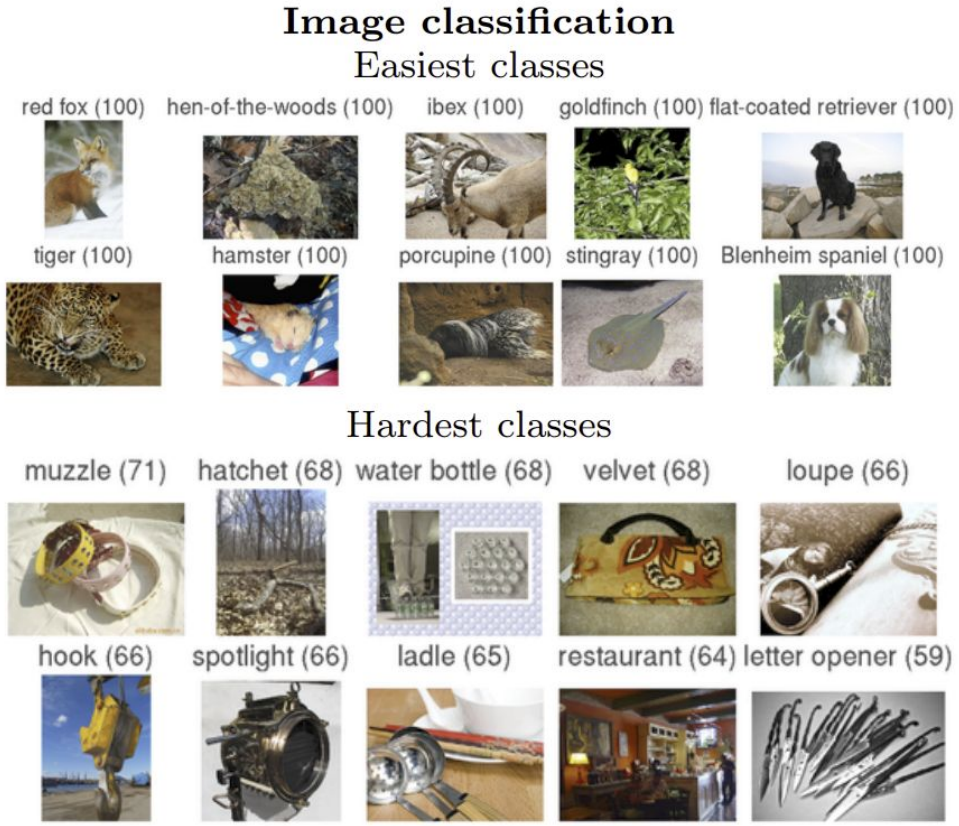
The History

- The image processing challenge (read the summary here)
- Started in 2010 to get good image processing algorithms
- Contestants in this competition have two simple tasks. Presented with an image of some kind,
  - The first task is to decide whether it contains a type of object or not. For example, a contestant might decide that there are cars in this image but no tigers.
  - The second task is to find a particular object and draw a box around it. For example, a contestant might decide that there is a screwdriver at a certain position with a width of 50 pixels and a height of 30 pixels.
  - Oh, and one other thing: there are 1,000 different categories of objects ranging from abacus to zucchini, and contestants have to scour a database of over 1 million images to find every instance of each object. Tricky!
- In 2012 a team from Canada improved on the algorithm and prediction
- Link of results here

SuperVision - 2012 and since

- SuperVision, for example, consists of some 650,000 neurons arranged in five convolutional layers.
- It has around 60 million parameters that must be fine-tuned during the learning process to recognize objects in particular categories.
- It is this huge parameter space that allows the recognition of so many different types of object.
- Since 2012, several groups have significantly improved on SuperVision's result.
- In 2014 year, an algorithm called GoogLeNet, created by a team of Google engineers, achieved an error rate of only 6.7 percent.
- Present day algorithms beat humans handily!

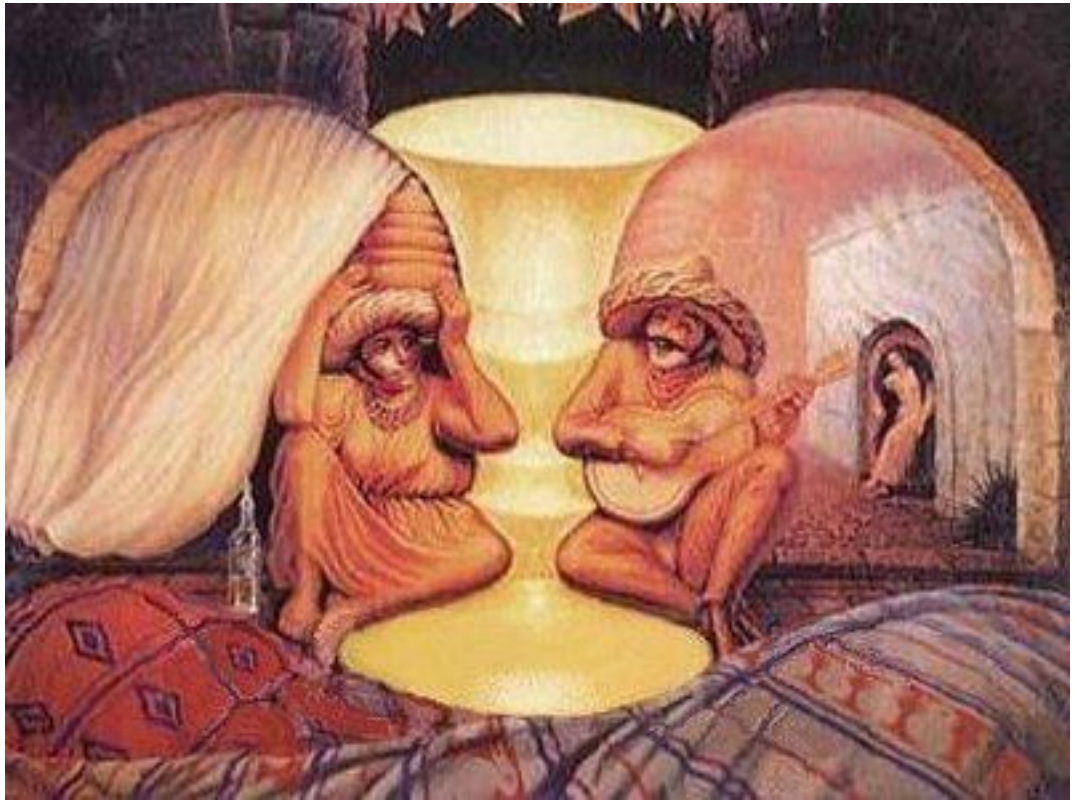
Image set for the 2012 challenge



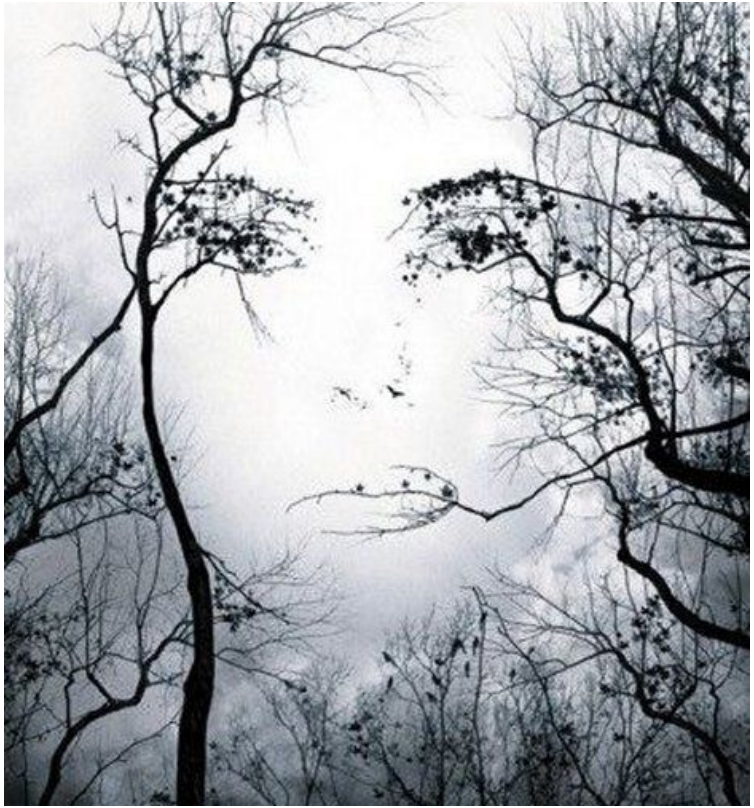
2012 - SuperVision

- This was the first time that a deep convolutional neural network had won the competition, and it was a clear victory.
- In 2010, the winning entry had an error rate of 28.2 percent,
- in 2011 the error rate had dropped to 25.8 percent.
- But SuperVision won with an error rate of only 16.4 percent in 2012 (the second best entry had an error rate of 26.2 percent).
- That clear victory ensured that this approach has been widely copied since then.

This is an Image of?



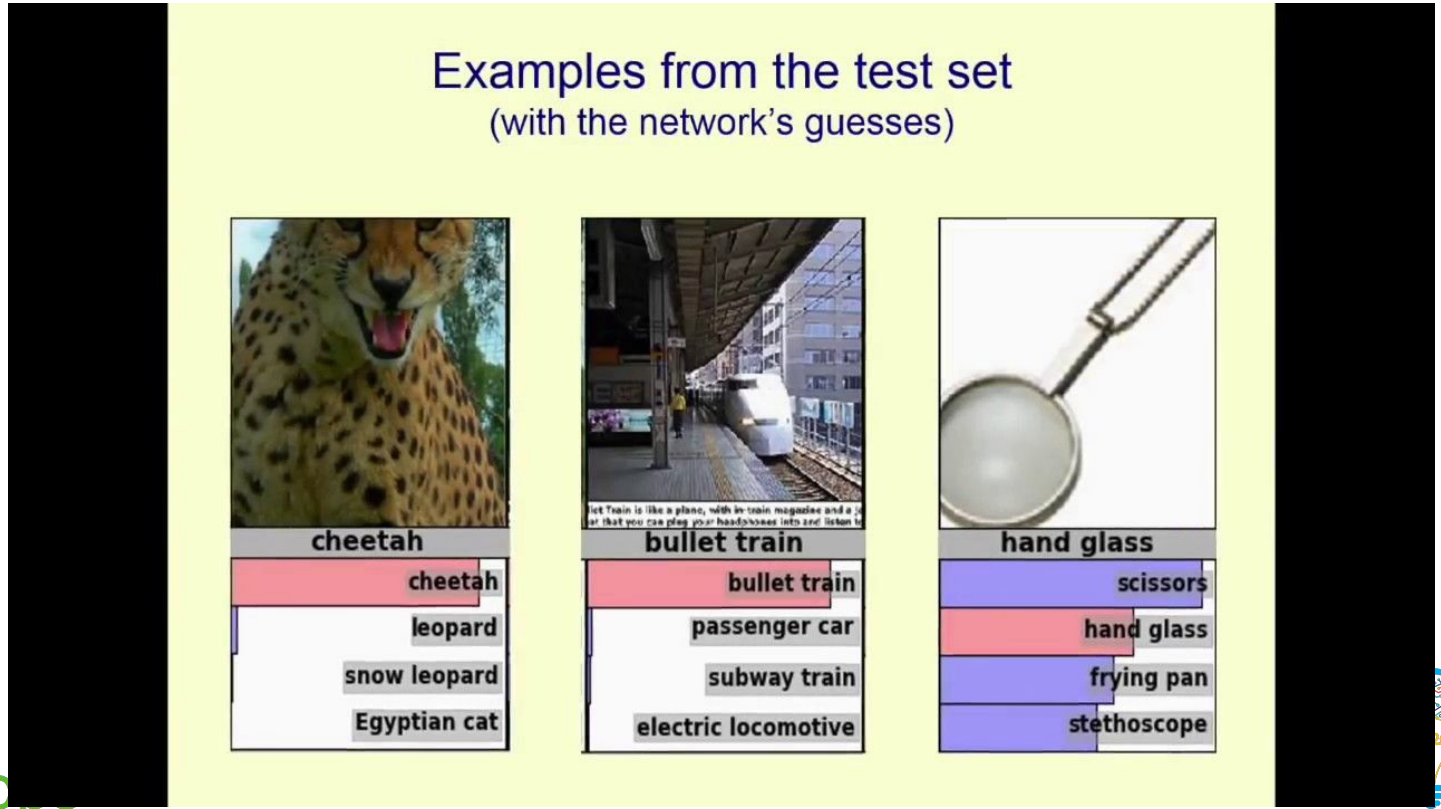
This is an Image of?



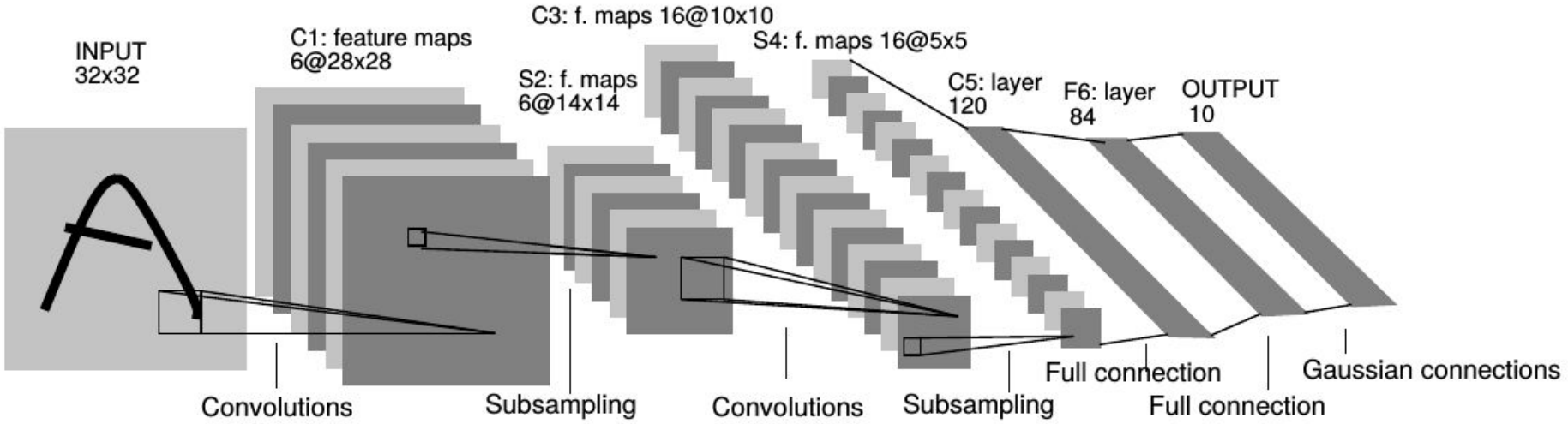
Comparison of Search Terms

https://trends.google.com/trends/explore?date=2015-07-15%202018-08-15&q=Convolutional%20Neural%20Network,Artificial%20Neural%20Network

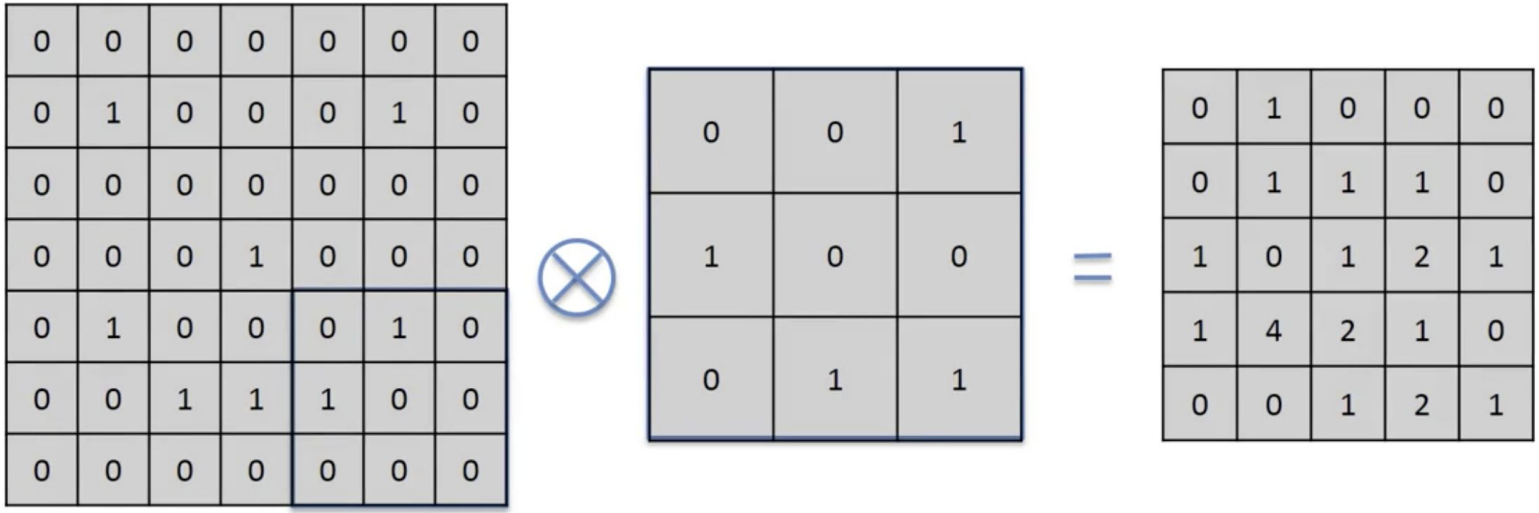
Hinton's example



Birth of CNN (Yann Lecun)



Convolution



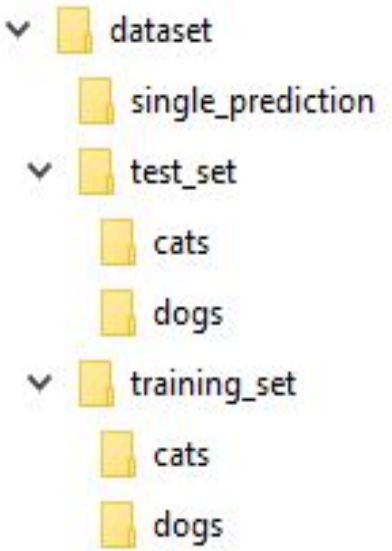
Gimp Image

https://docs.gimp.org/en/plug-in-convmatrix.html

Seven Step Process

1) Arrange Images in Folders (by Labels)
2) Apply Convolutions
3) Pooling
4) Flattening
5) ANN Layer
6) Activation Function
7) Collecting Prediction

1 - Folder Structure (Data Preprocessing)

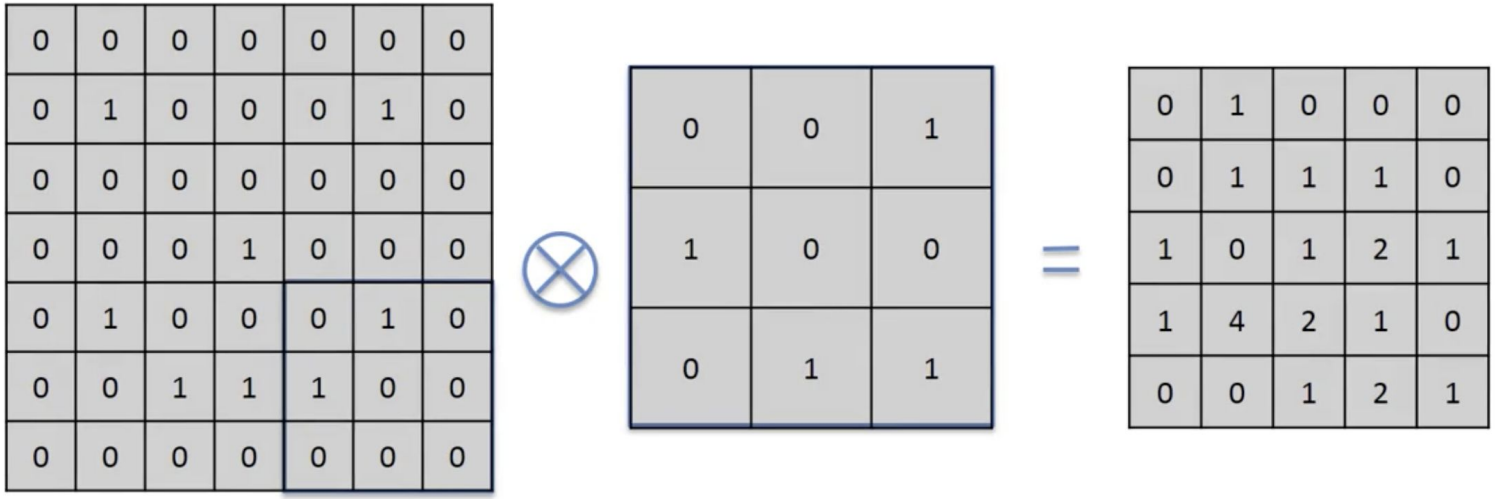


In our example the dataset is 10000 images of cats and dogs. 80:20 split is applied where train set contains 8000 images and test set contains 2000 images.

Step 2: Convolution

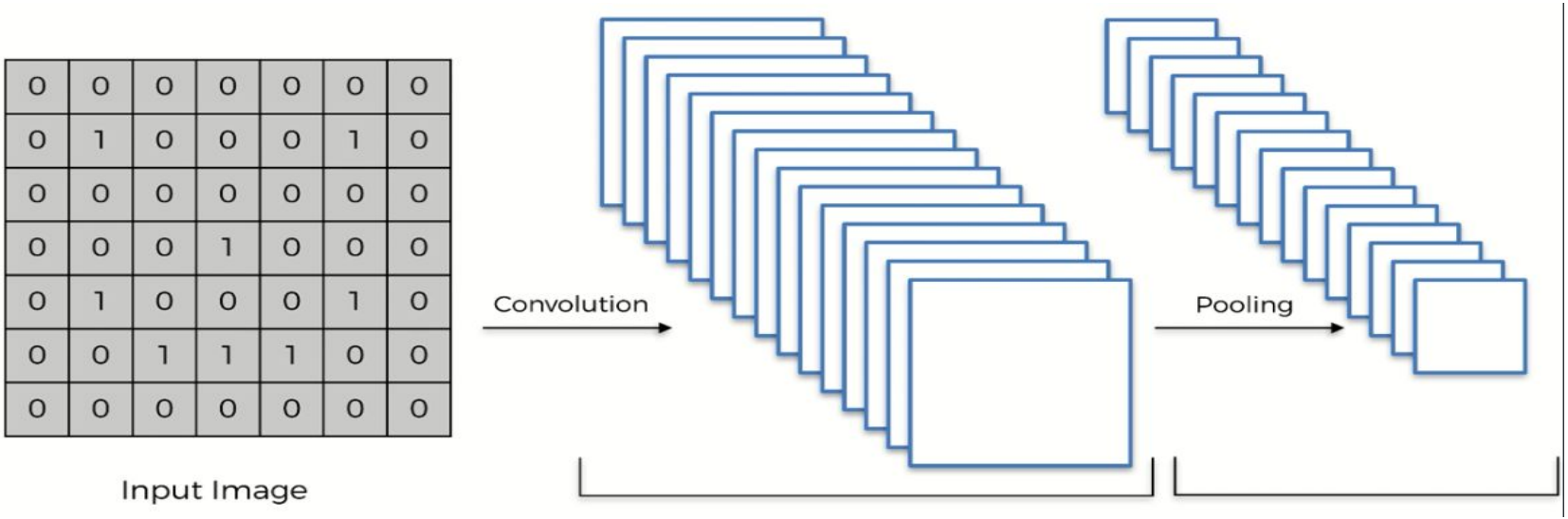
- We have an input image, we convert the image into table of pixel values.Several features are applied to the input image in the form of feature detectors.
- Then we move through the image to match these features with the image, whenever a match is found we get an feature map with higher values.
- For each feature detector we apply we get feature map of different values where higher values represent a unique feature found by the feature detector.
- In this step we choose the number of feature detectors to get feature maps.

Convolution Explained

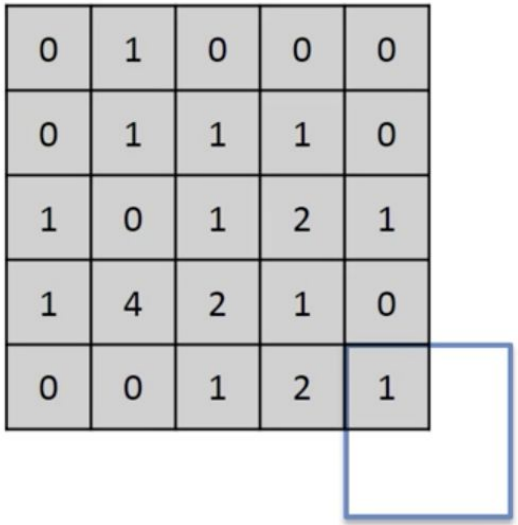


Step 3: Pooling

```
classifier.add(MaxPooling2D(pool_size = (2, 2)))
```



Pooling Explained



Feature Map

Max Pooling

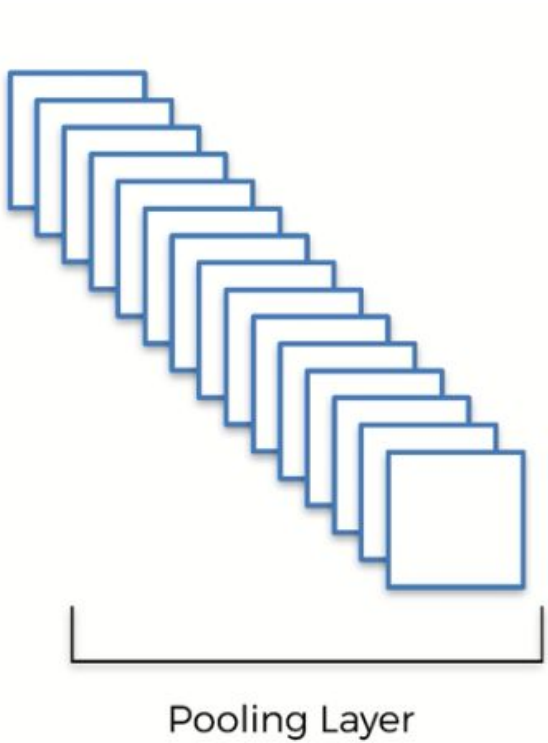
1	1	0
4	2	1
0	2	1

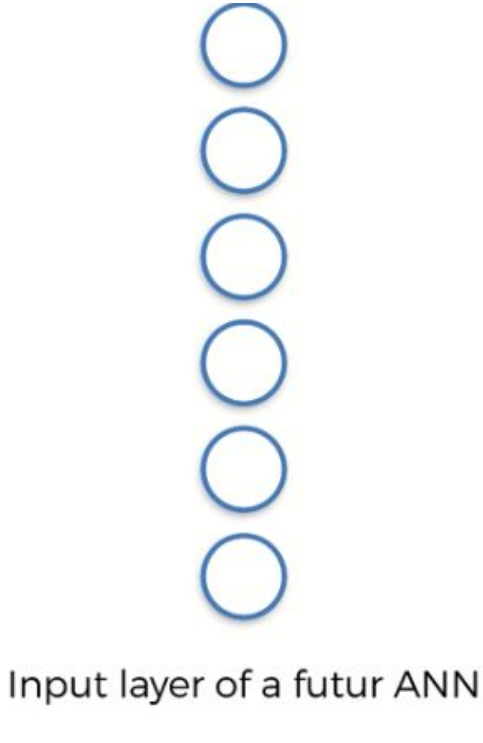
Pooled Feature Map

Step 3: Pooling

- It consists of reducing the size of feature maps.
- We slide over the feature map a pooling matrix to reduce the size of the feature map by half of its size+1 it it is initially an odd ordered matrix and half of the size if it is initially an even ordered matrix.
- For ex: if the initial size is 5 by 5 and we slide a 2 by 2 matrix with a stride two then we get a pooled matrix of size 3 by 3.
- We do this so that we reduce the number of nodes when we perform flattening. Since we build a one dimensional matrix corresponding to each feature map, hence reducing the number of nodes in fully connected layer and thus reducing time and space complexity.

Step 4: Flattening





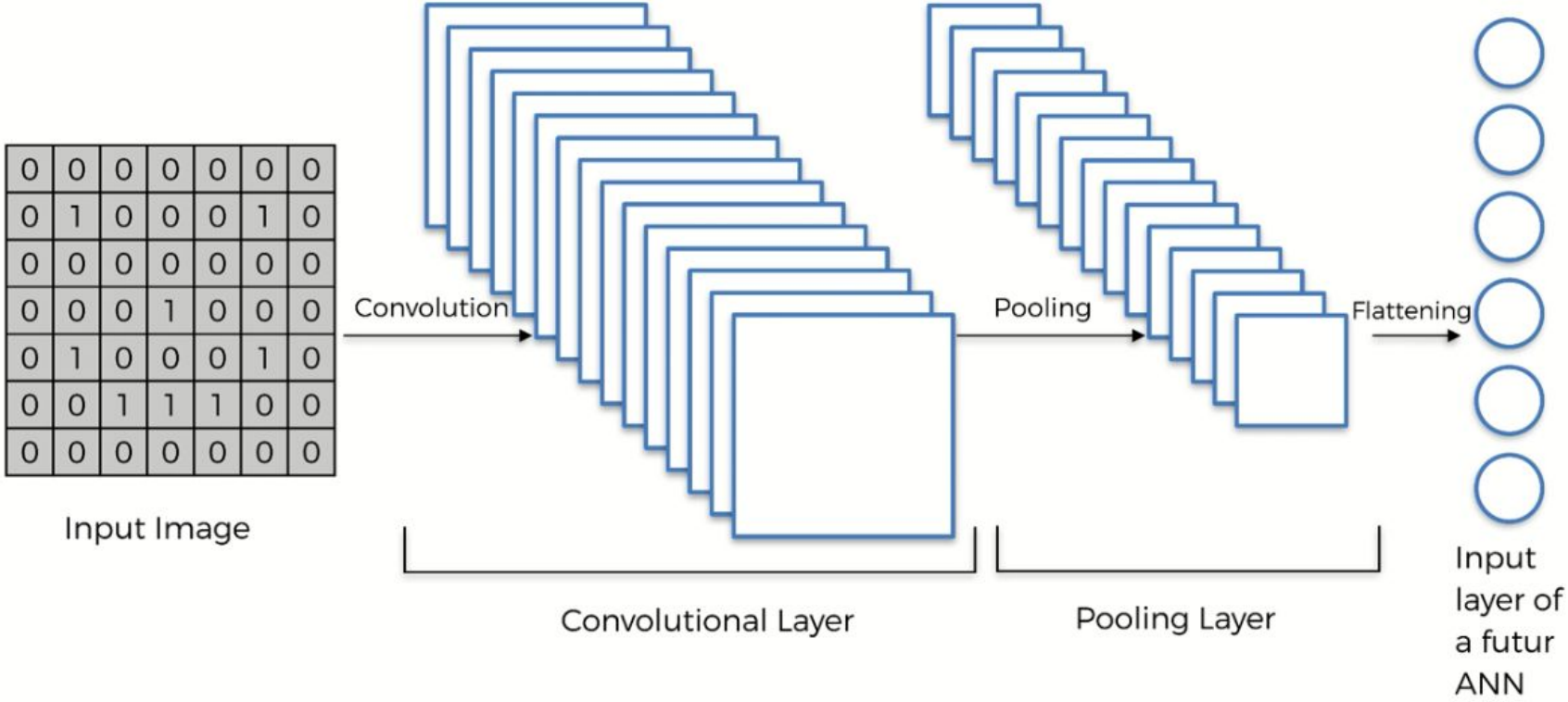
Step 4: Flattening

In this we consider feature maps of all the different features and we build a single dimensional vector which is huge and is going to be served as input layer for neural network.

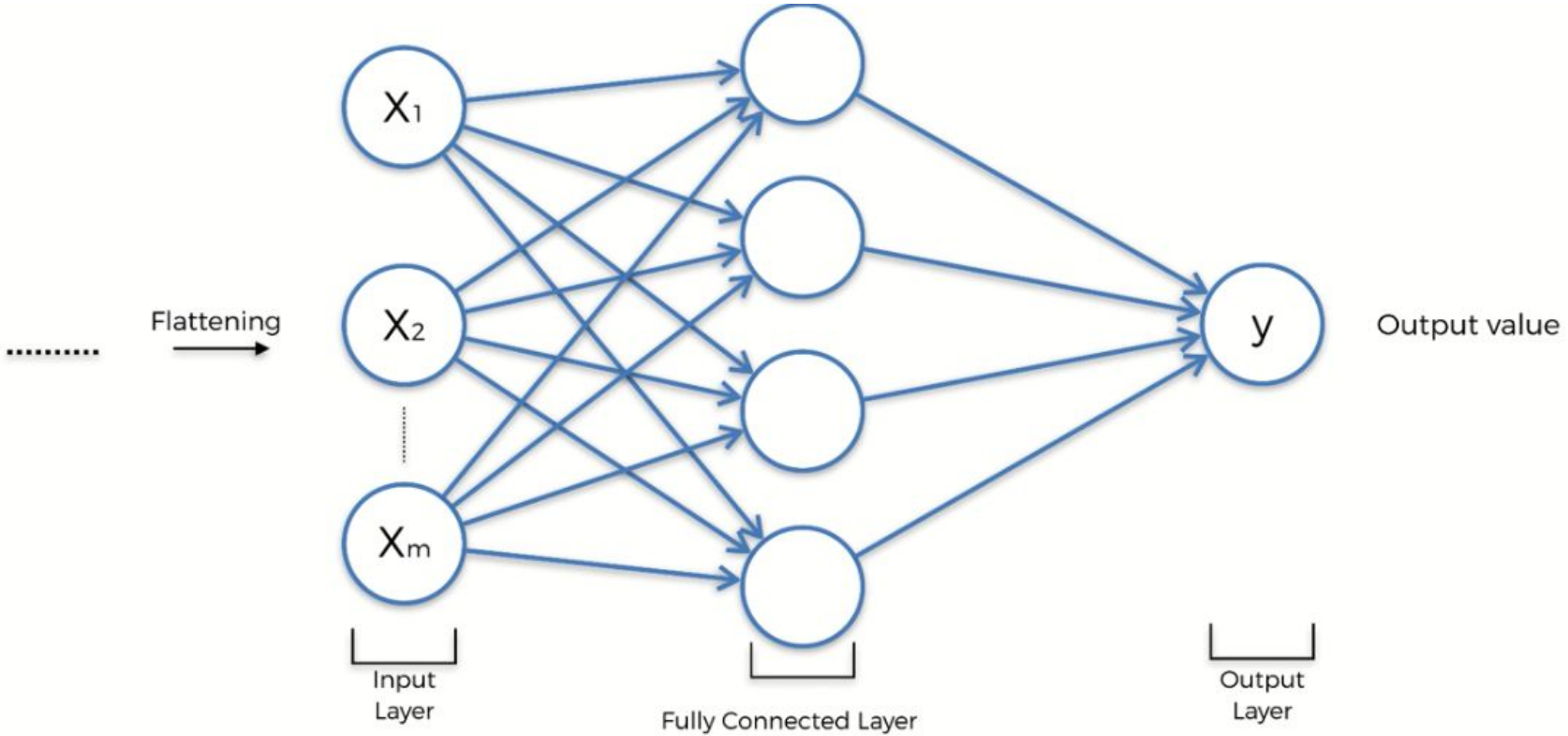
We can directly perform flattening over the image but this leads to representation of a single dimension vector where each value represents a single node without relation with other nodes.

```
classifier.add(Flatten())
```

Steps completed till now



Step 5: Connect to ANN

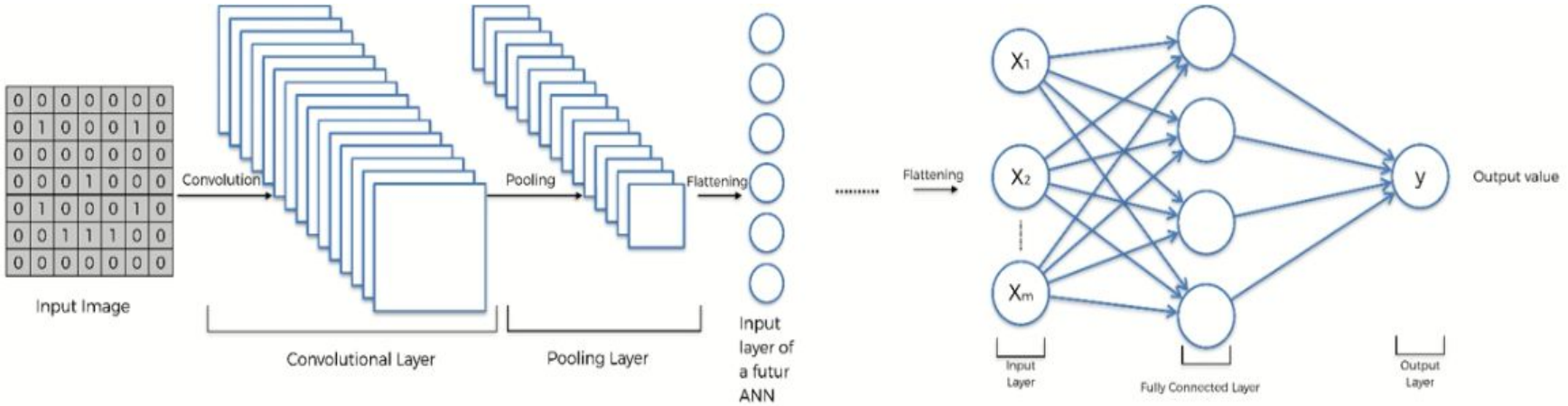


Step 5: Full Connection

```
classifier.add(Dense(units = 128, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))
```

- First Dense Function (Hidden layer):
  - The units represents the number of nodes in the hidden layer.
  - We have to activate the nodes in the hidden layer so that they can vote towards a signal passing through them. For this we use rectifier activation function called relu.
- Second Dense Function (Output Layer):
  - The units contain only 1 since the output layer contains only one node.
  - We use sigmoid function since we have to get binary output (cats or dogs).

Whole CNN


Input Image
Convolution
Pooling
Flattening
Convolutional Layer
Pooling Layer
Input layer of a futur ANN
Flattening
X1
X2
Xm
y
Output value
Input Layer
Fully Connected Layer
Output Layer

Step 6: Activation Function

1. Sigmoid
2. Softmax
3. Cross Entropy

Step 7: Verifying Data

Will be completed